

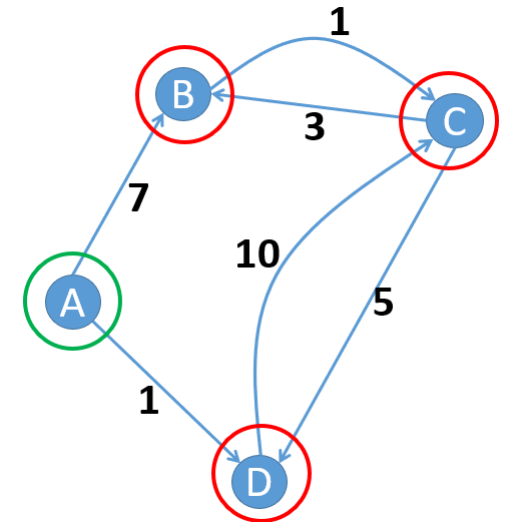
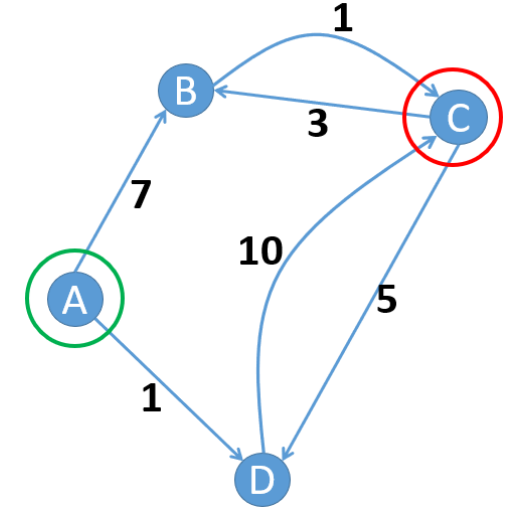
# Shortest path planning on grids and graphs using networks without and with learning



**Tomas Kulvicius**

# Planning problem

- **Planning:**
  - Search for a sequence of actions that brings agent/world from an initial state to one or more goal states
- **Shortest path problem:**
  - Find shortest path between two vertices in a graph such that sum of weights of the path edges is minimised
- **Single-source shortest paths (SSSP) problem:**
  - Find shortest paths from one vertex (source) to all other vertices in a graph



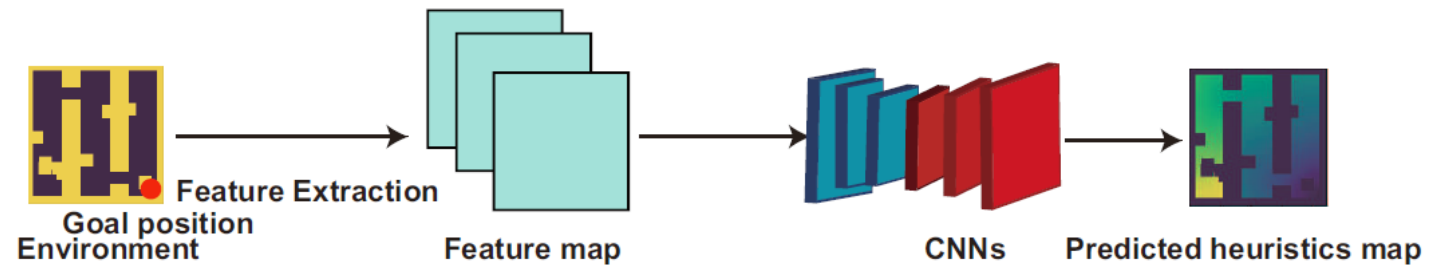
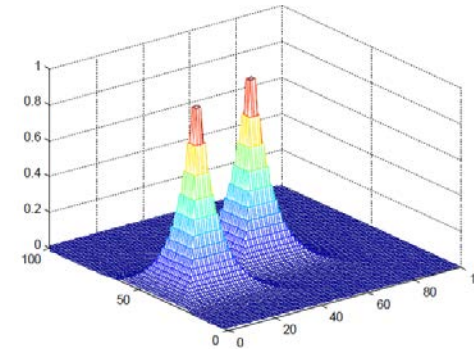
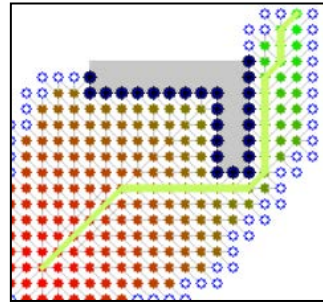
# Applications of SSSP algorithms

- **Computer networks** (shortest path between two computers)
- **Social networks** (shortest path between two persons)
- **Trading and financy**
- **Multi-agent systems** (games)
- **Path/task planning** (routing, robotics)



# State-of-the-art

- **Conventional methods**
  - BFS, Dijkstra, Bellman-Ford
- **Biologically inspired**
  - TD(0): Reinforcement learning
  - Activity propagation net (AP-Net)
- **Deep learning (DL)**
  - DMLP
  - FCN
  - LSTM
  - Deep RL

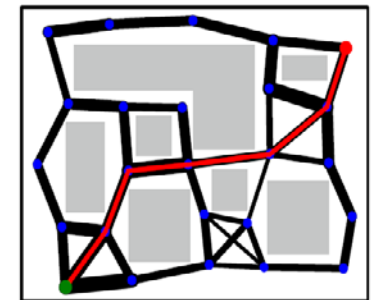
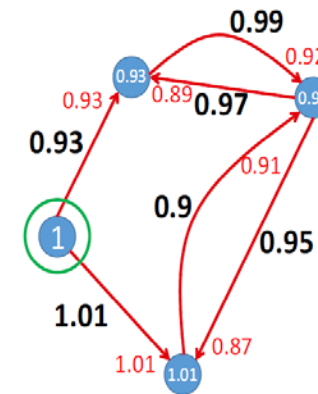
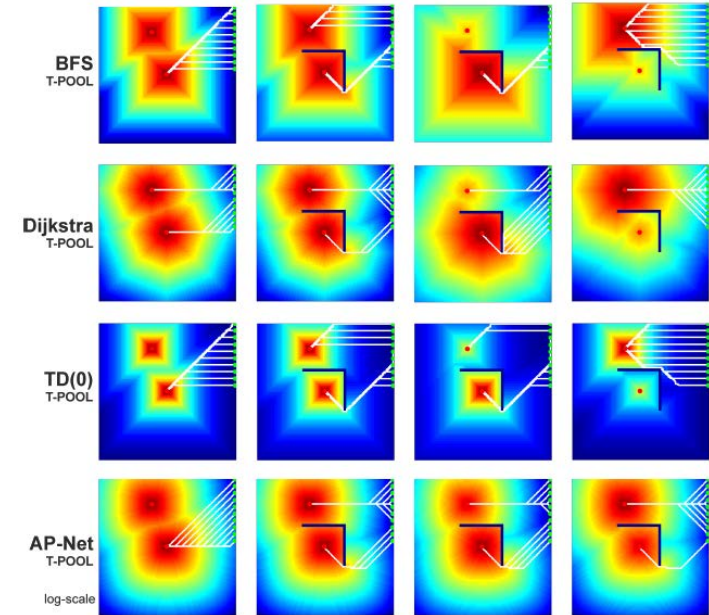


- Iterative methods
- DL approaches require data and learning
- DL approaches do not guarantee optimal solutions



# Contribution

- **Part I: (Multi-Source) Path planning on grids**
  - Deep network without learning
  - Solution in one forward-pass
- **Part II: Path planning on graphs**
  - Neural implementation of Bellman-Ford algorithm
  - Combining optimal path planning with Hebbian learning

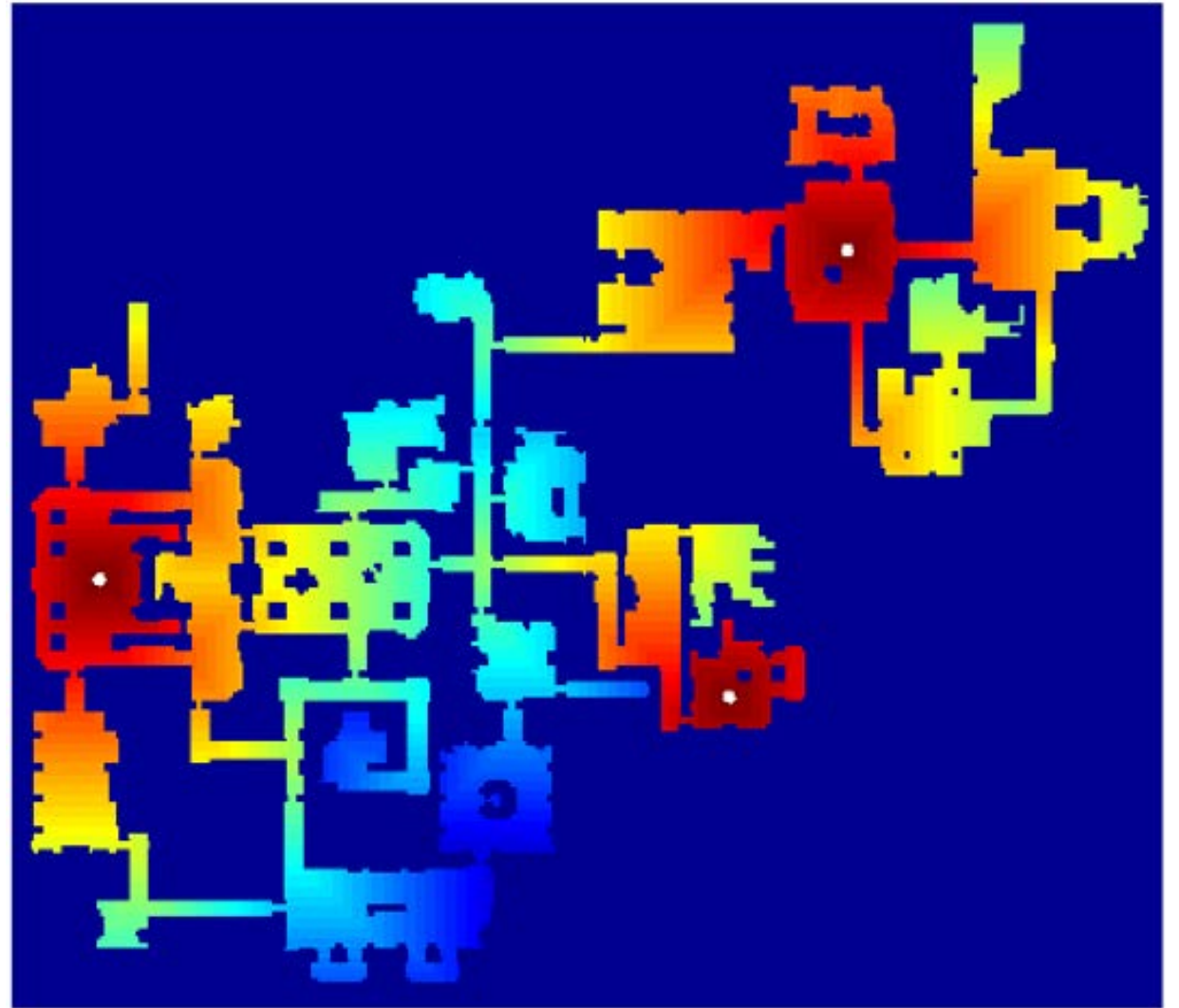


# Part I: (Multi-source) Path planning on grids without learning

# Task definition

- Solve **SSSP\*** problem for multiple (un) weighted sources on a grid representation
- Generate activity map
- Find paths to the closest source by following gradient

\***SSSP** – Single-Source Shortest Path



# Network architecture: T-POOL

**BFS**<sub>[max-pool]</sub>

max **X**

**TD(0)**<sub>[scale + max-pool]</sub>

max

0.9	0.9	0.9
0.9	0.9	0.9
0.9	0.9	0.9

◦ **X**

**APN**<sub>[scale + avg-pool]</sub>

avg

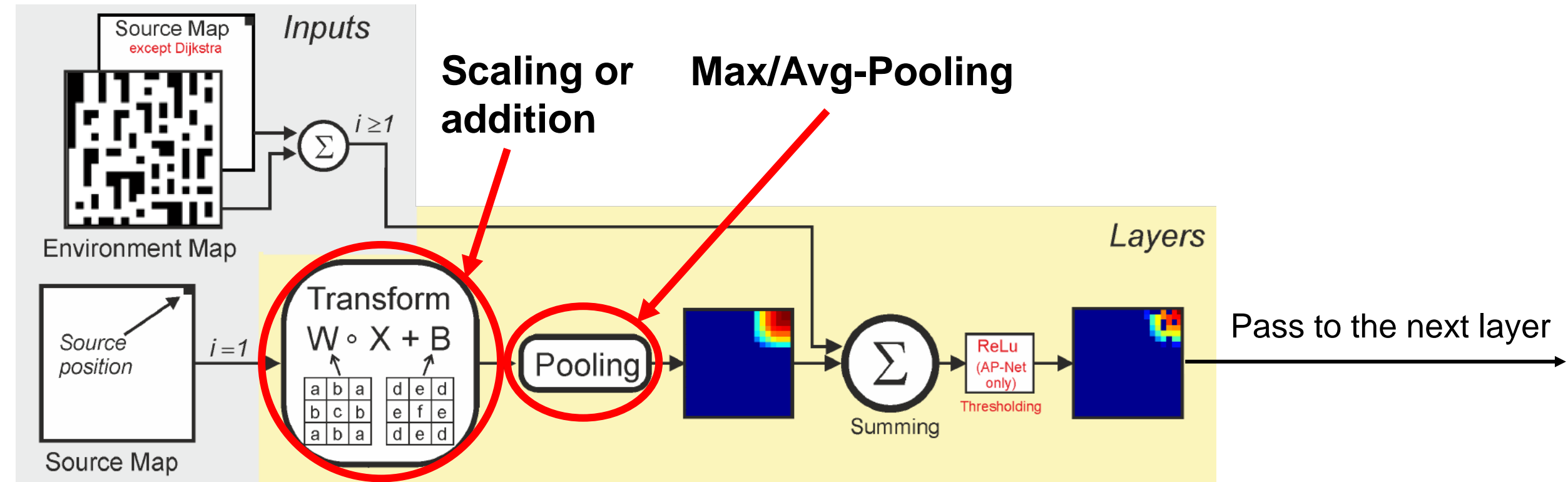
$1/\sqrt{2}$	1	$1/\sqrt{2}$
1	0	1
$1/\sqrt{2}$	1	$1/\sqrt{2}$

◦ **X**

**Dijkstra**<sub>[add + max-pool]</sub>

max **X** +

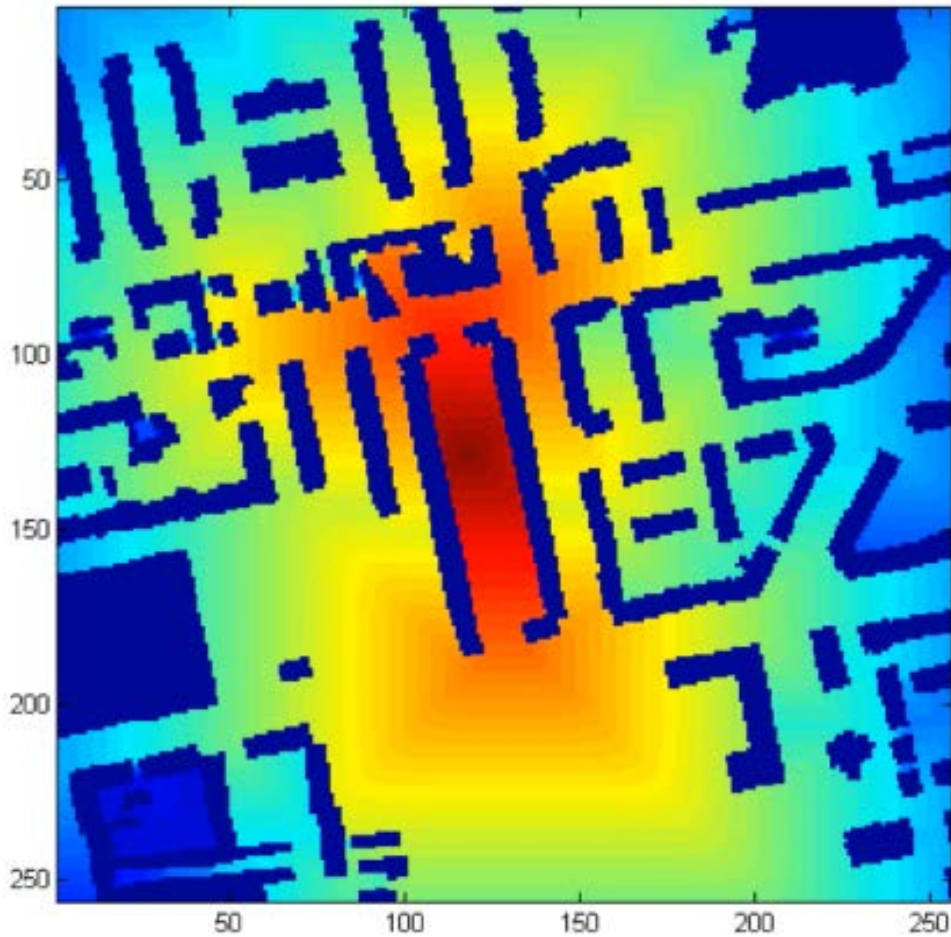
$-\sqrt{2}$	-1	$-\sqrt{2}$
-1	0	-1
$-\sqrt{2}$	-1	$-\sqrt{2}$



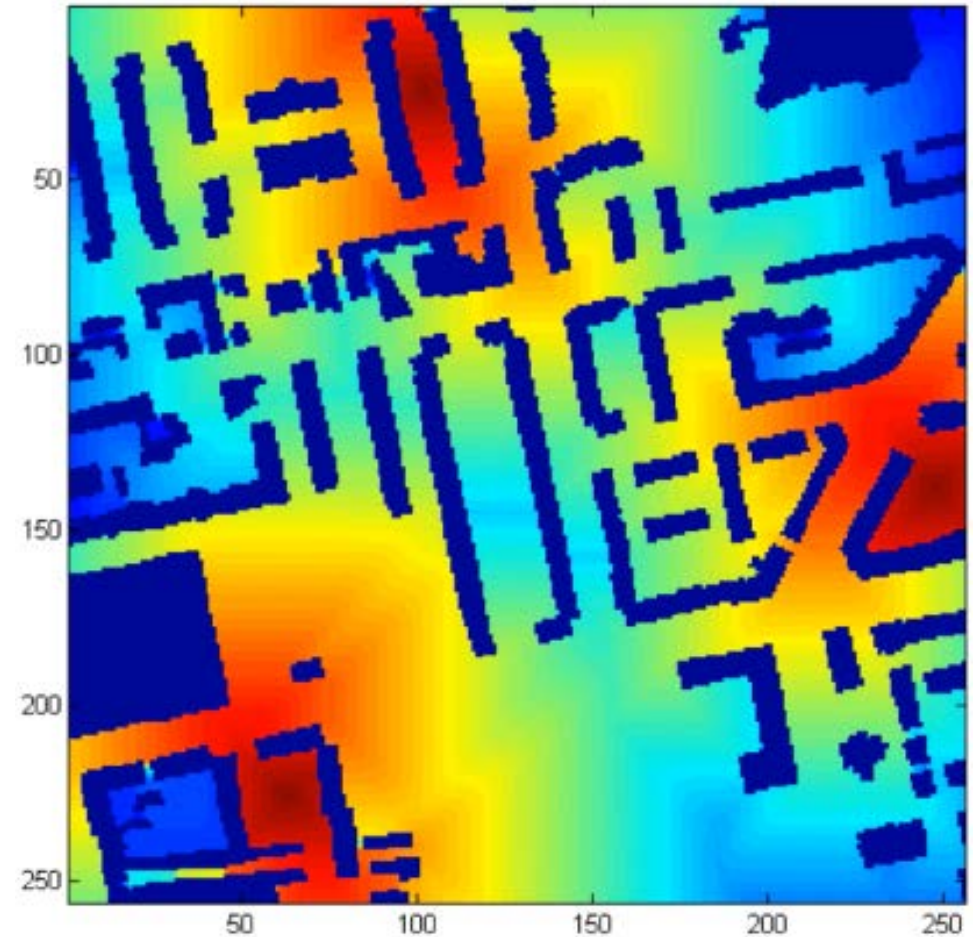


# Example: T-POOL-BFS

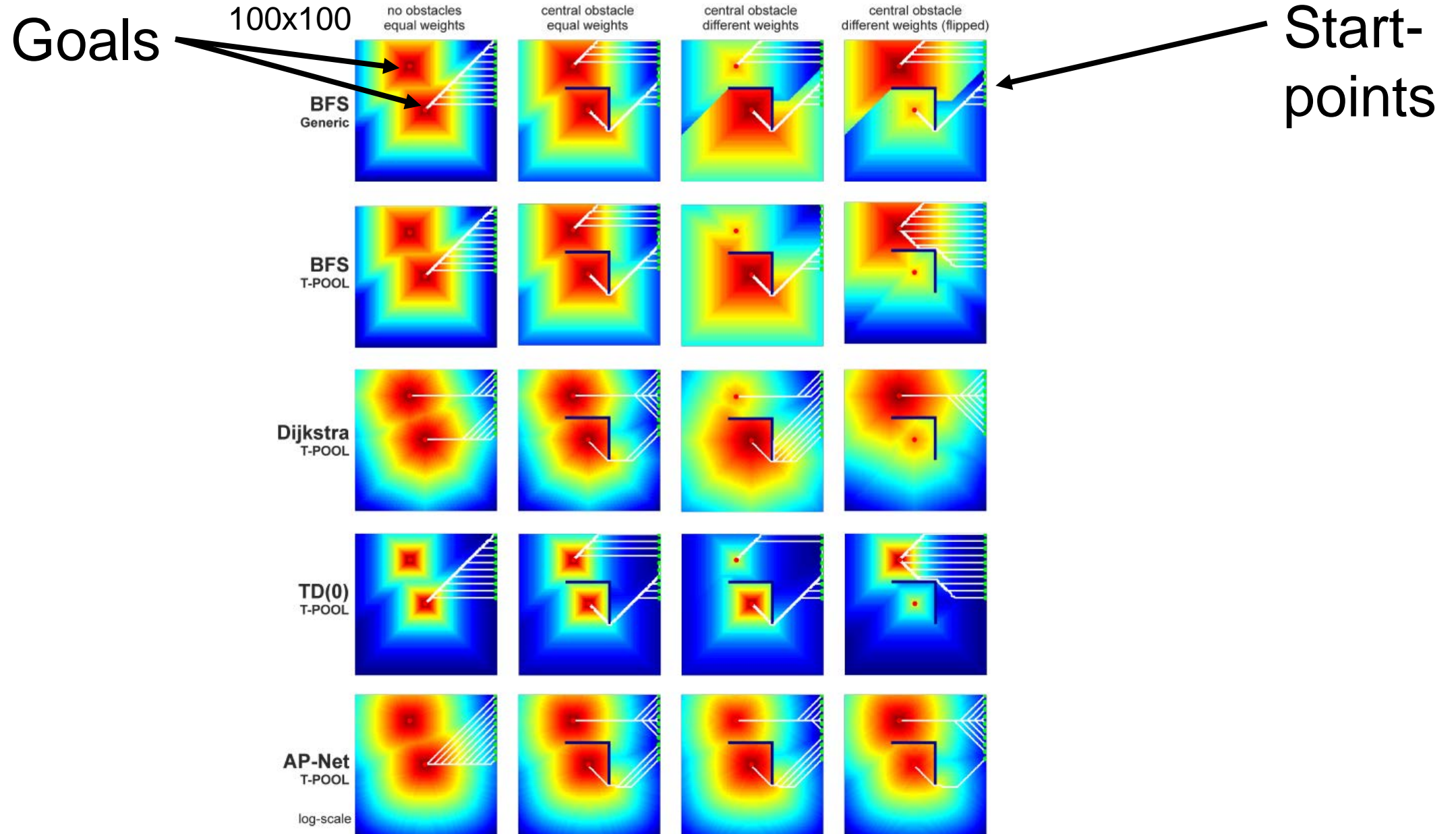
Single source



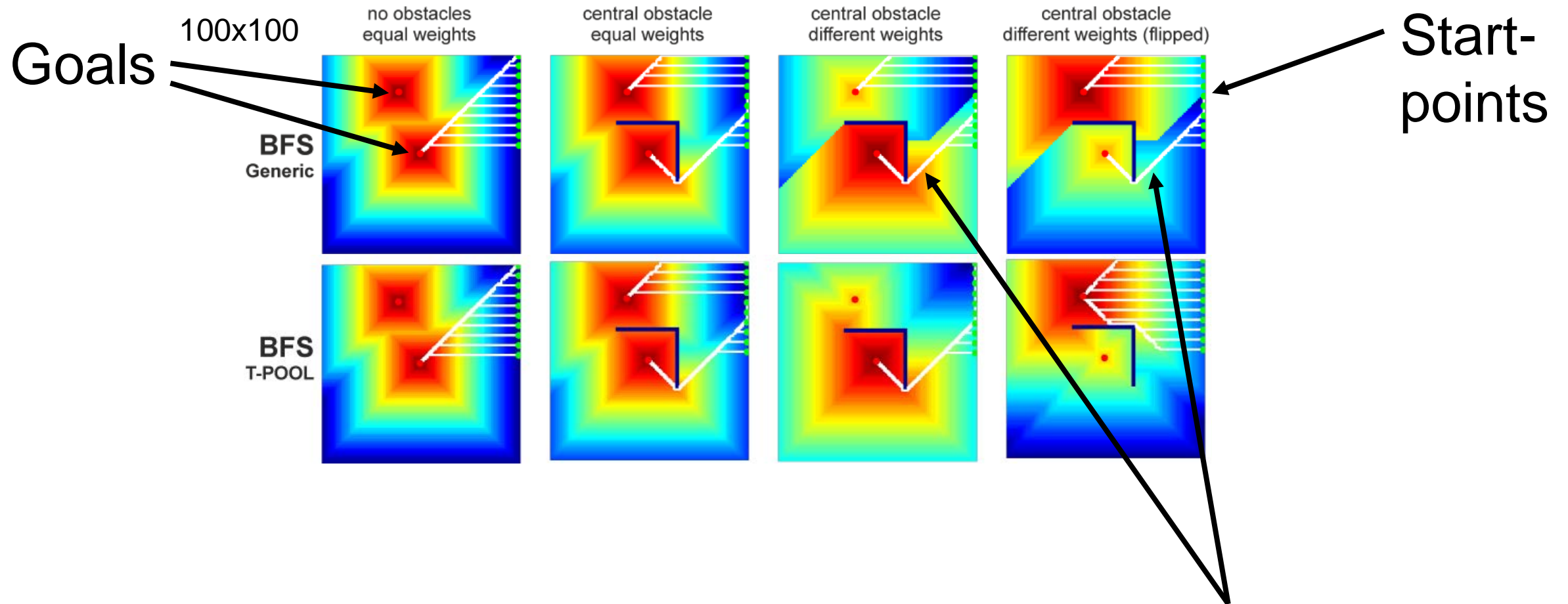
Three sources



# Emulation of BFS, TD(0), Dijkstra, and AP-Net



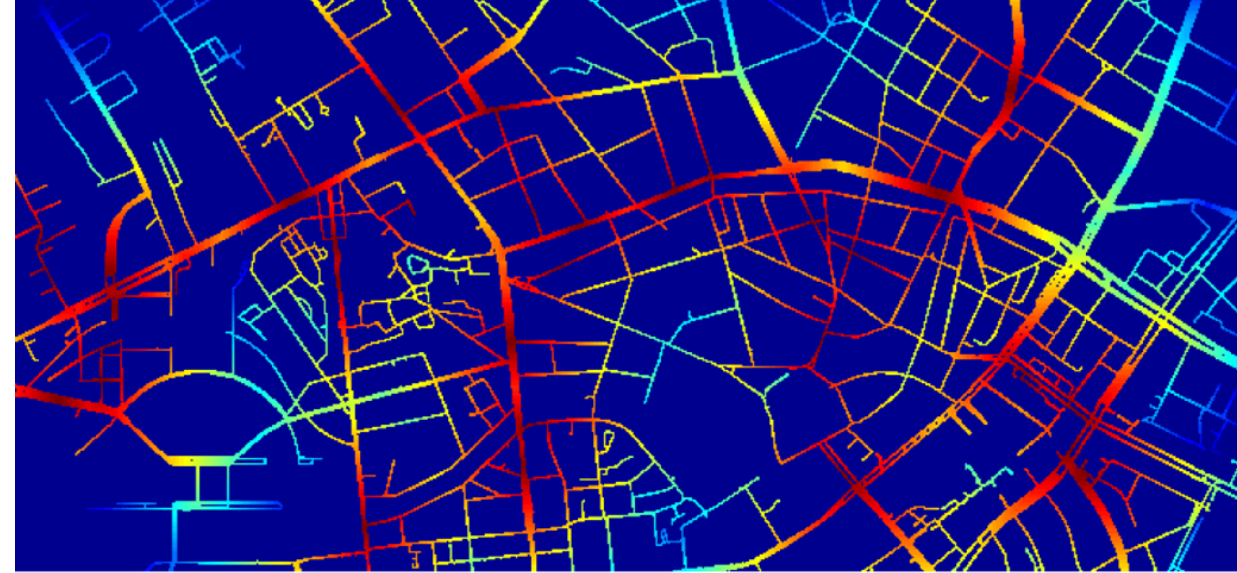
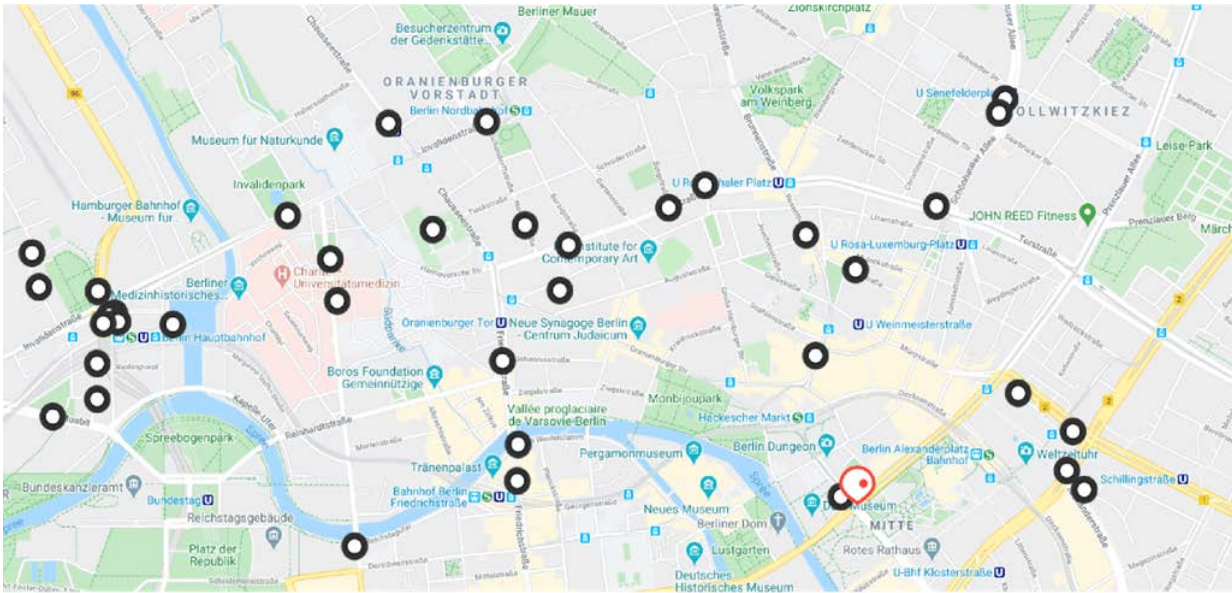
# Generic BFS vs. T-POOL-BFS



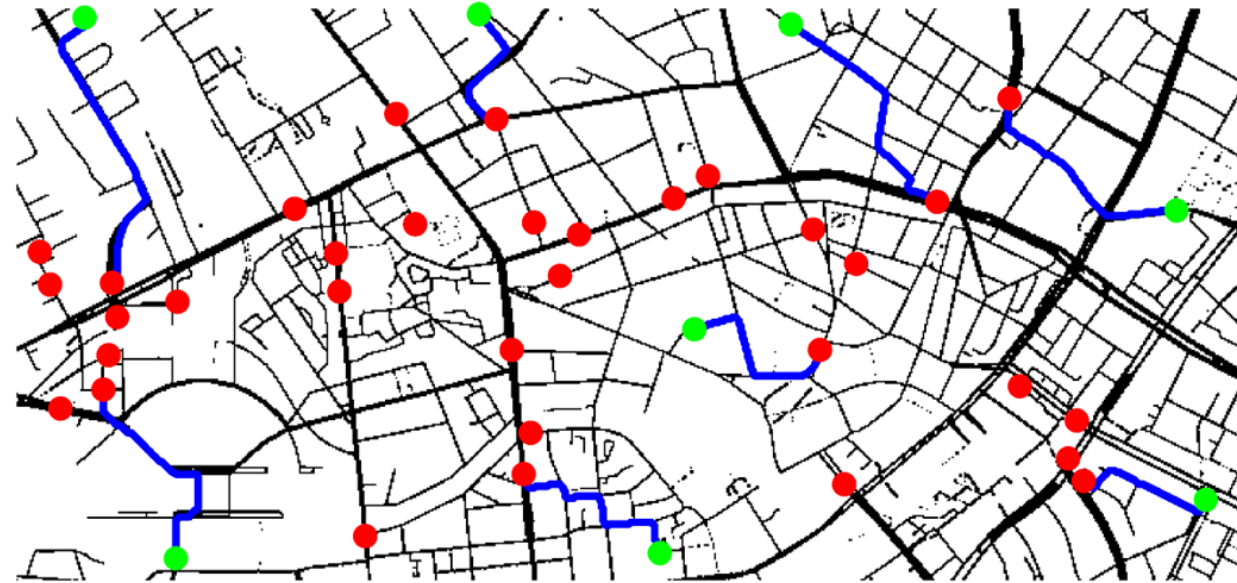
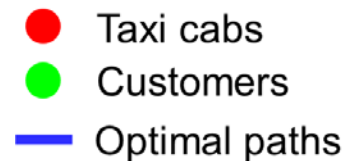
Generic BFS does not work in case of multiple weighted sources!



# Application to taxi scenario



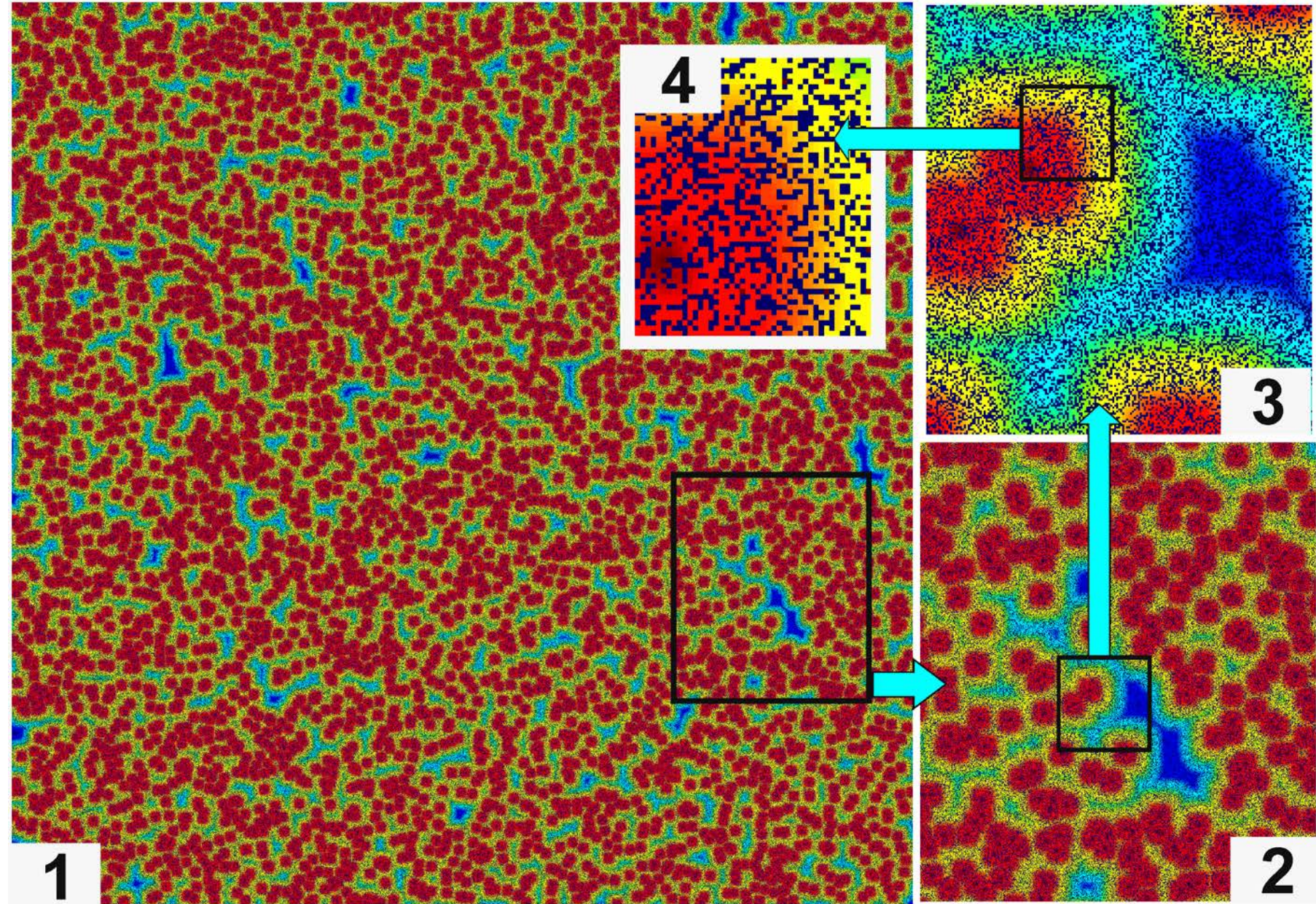
- Berlin city map (332x709)
- 33 taxi cabs (sources)
- T-POOL-BFS,  
160 layers, **14.1 ms**





# Solving large grids

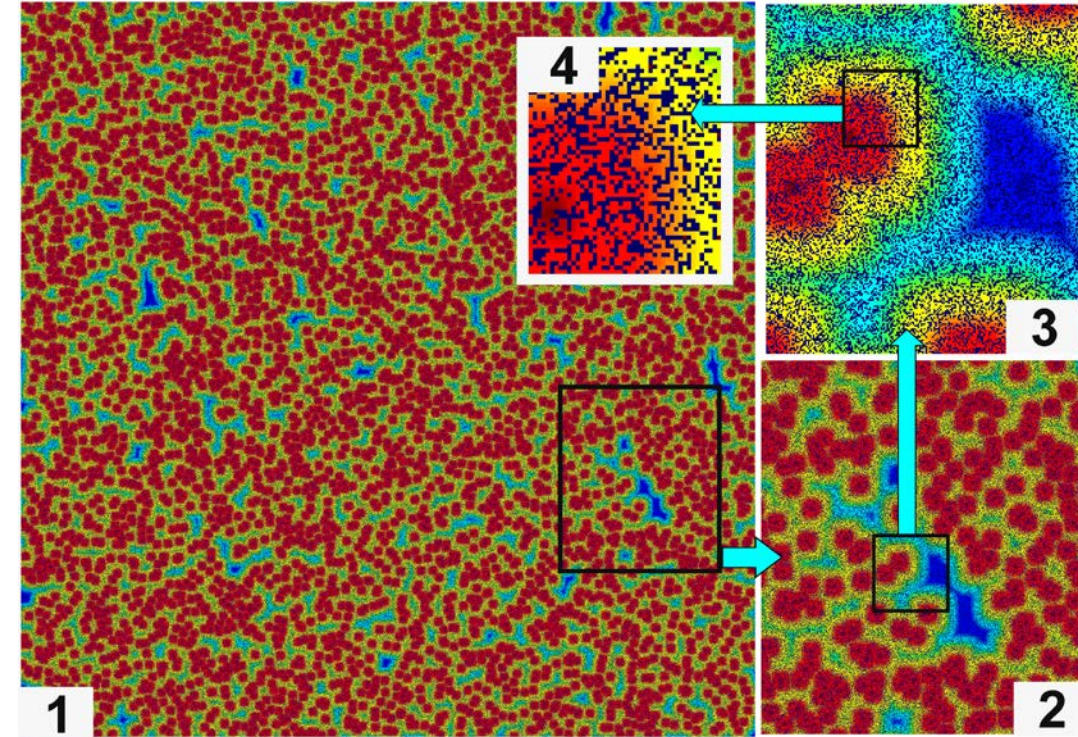
- Random maze  
( $n=26,000$ ;  
vertices= $676 \cdot 10^6$ ;  
edges= $5.4 \cdot 10^9$ )
- **5,000** sources
- T-POOL-BFS, 650  
layers, **111.7 s**





# Deep networks without learning

- **Pros**
  - No training data and no training needed
  - Different algorithms can be emulated
  - Optimal solutions
  - Scalability
- **Cons**
  - Only uniform weights
  - Only for 2D or 3D grid structures

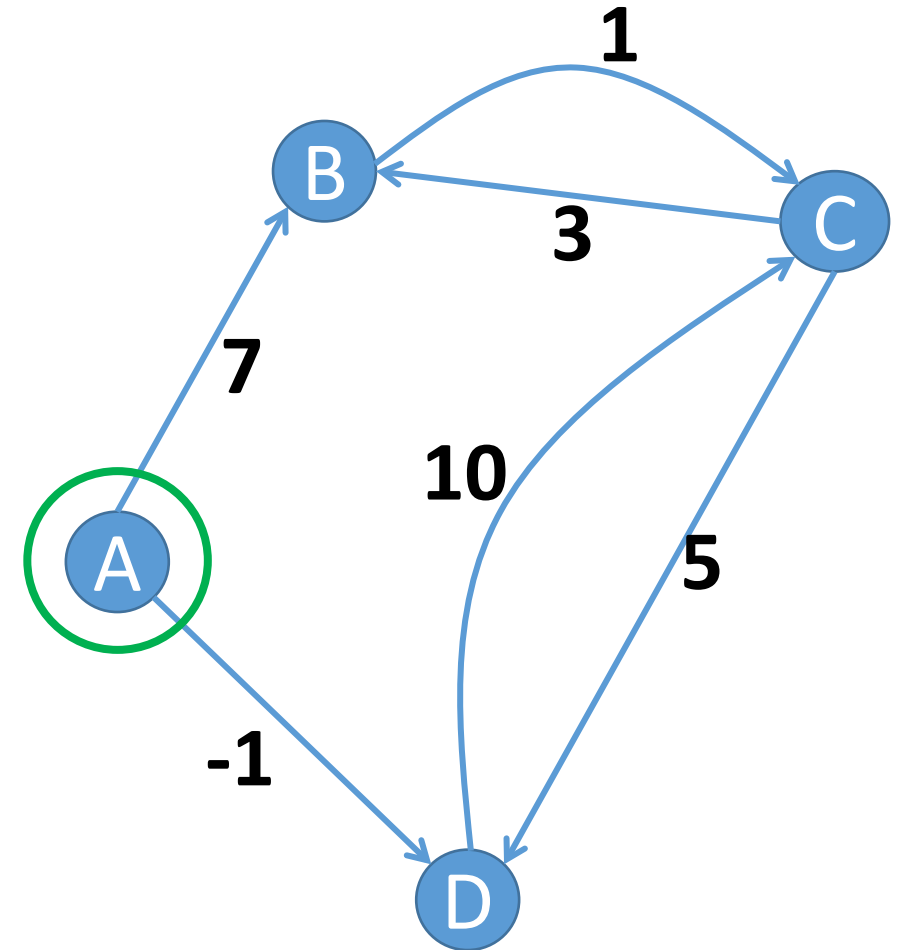


Can we solve weighted graphs with arbitrary weights?

# Part II: Path planning on graphs

# Task definition

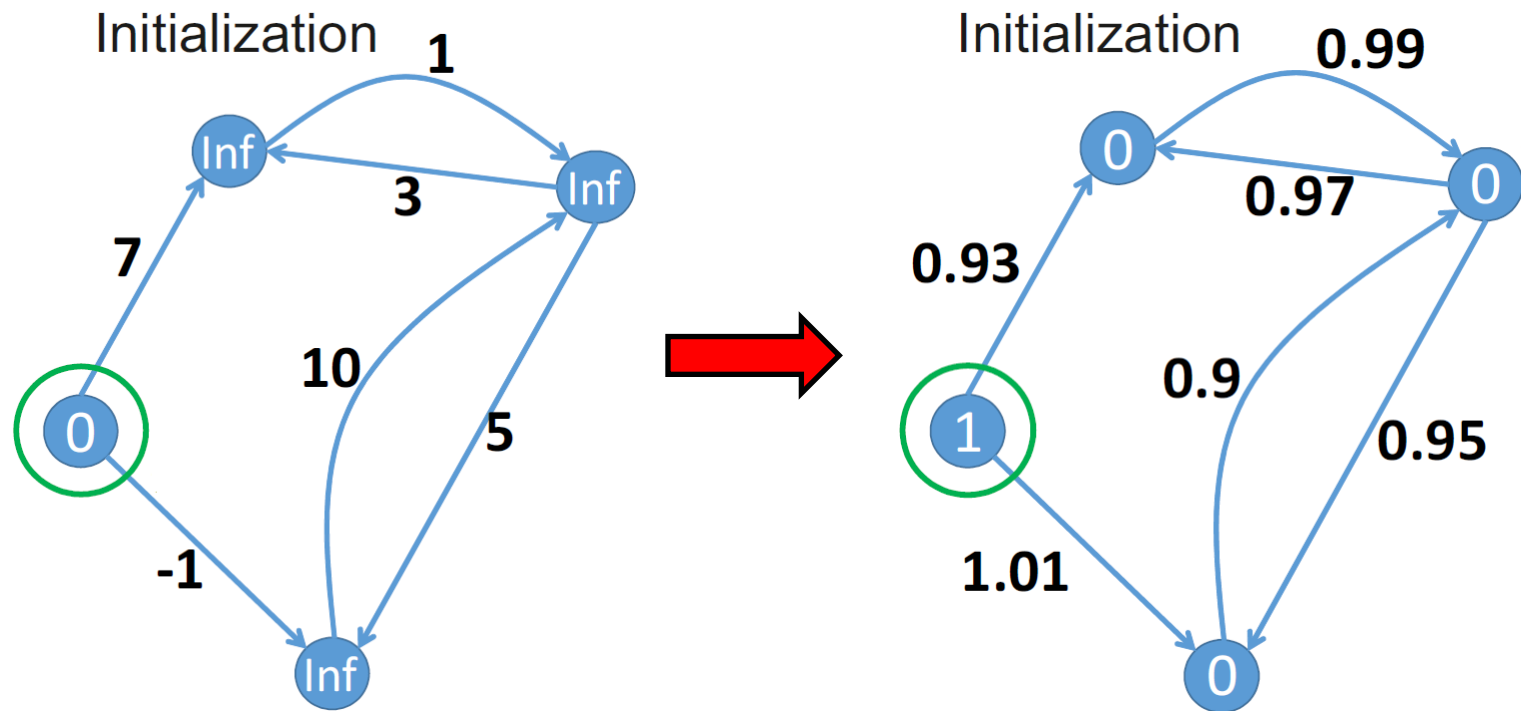
- Solve SSSP for directed graphs with arbitrary weights
- Conventional approach: Bellman-Ford (BF) algorithm
- Neuronal implementation of BF algorithm



# Neural network – Bellman-Ford (NN-BF)

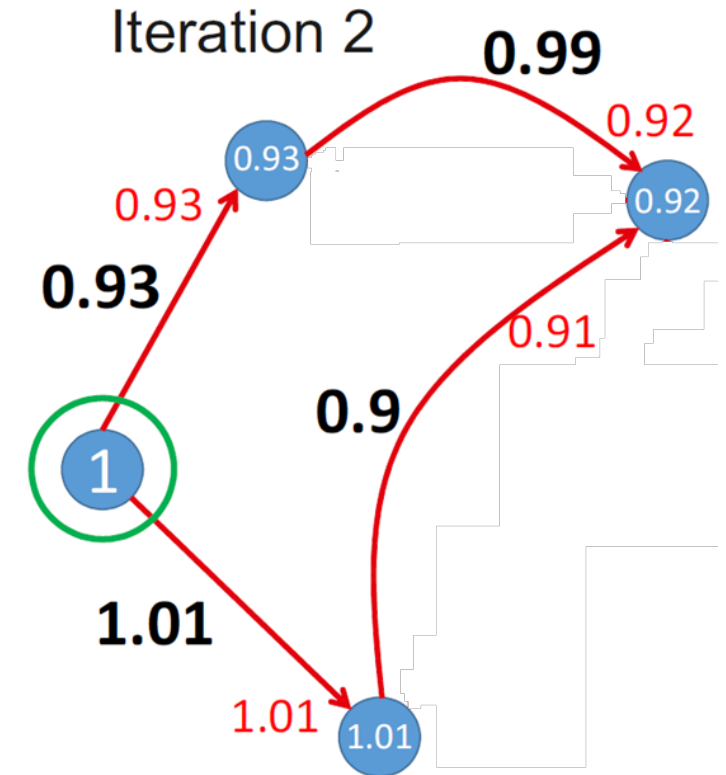
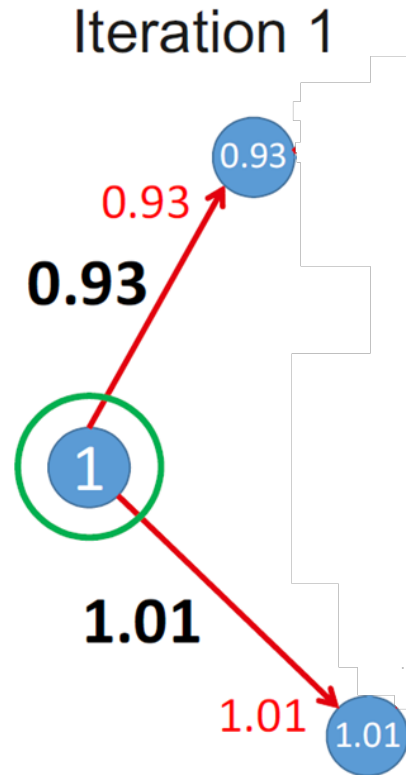
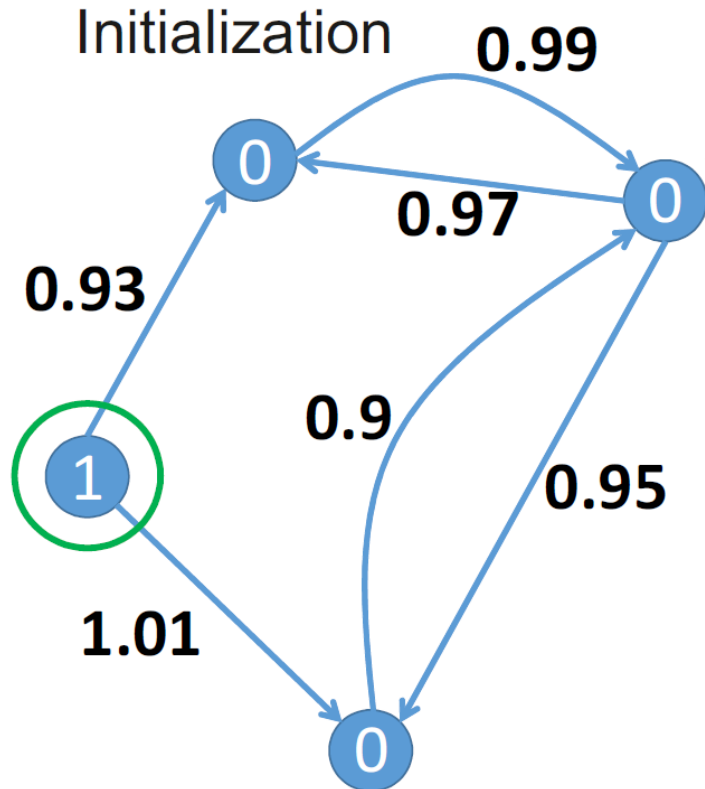
- **We invert weights:**  
Low cost → Large weight  
High cost → Small weight

$$w_{ij} = 1 - c_{ij}/K \text{ (e.g., } K=100\text{)}$$



# Neural network (NN-BF)

- Activity of node is:  $a_j = \max\{w_{ij} a_i\}$





# Optimality and algorithmic complexity

- **NN-BF is optimal**
  - $w_{ij} = 1 - c_{ij}/K$  ( $K \rightarrow \text{Inf}$ )
- **Bellman-Ford algorithm**
  - Worst case:  $O(|V| |E|)$
  - Best case:  $O(|E|)$
- **Neural Network**
  - Worst case:  $O(|V|^2 |E|/|V|) = O(|V| |E|)$
  - Best case:  $O(|E|)$

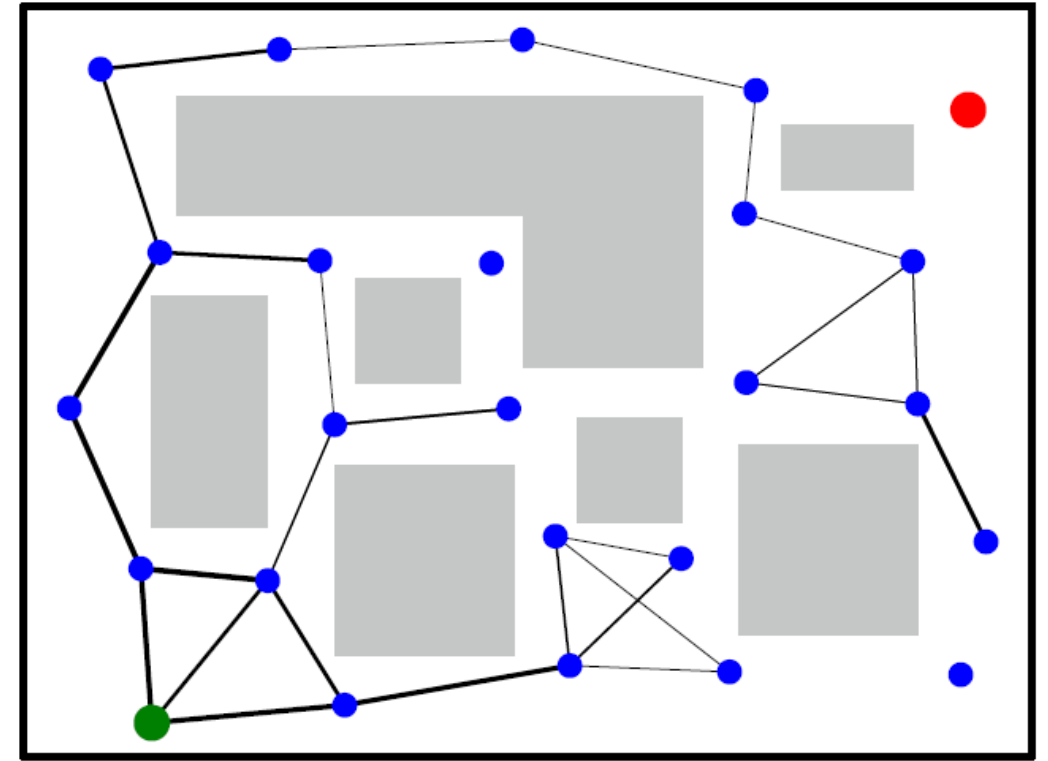
$ V $ - number of vertices $ E $ - number of edges
---

Kulvicius, T., Tamosiunaite, M., & Wörgötter, F. (2023). Combining optimal path search with task-dependent learning in a neural network. *IEEE Transactions on Neural Networks and Learning Systems*. DOI:10.1109/TNN.

# Combining planning and task-dependent learning

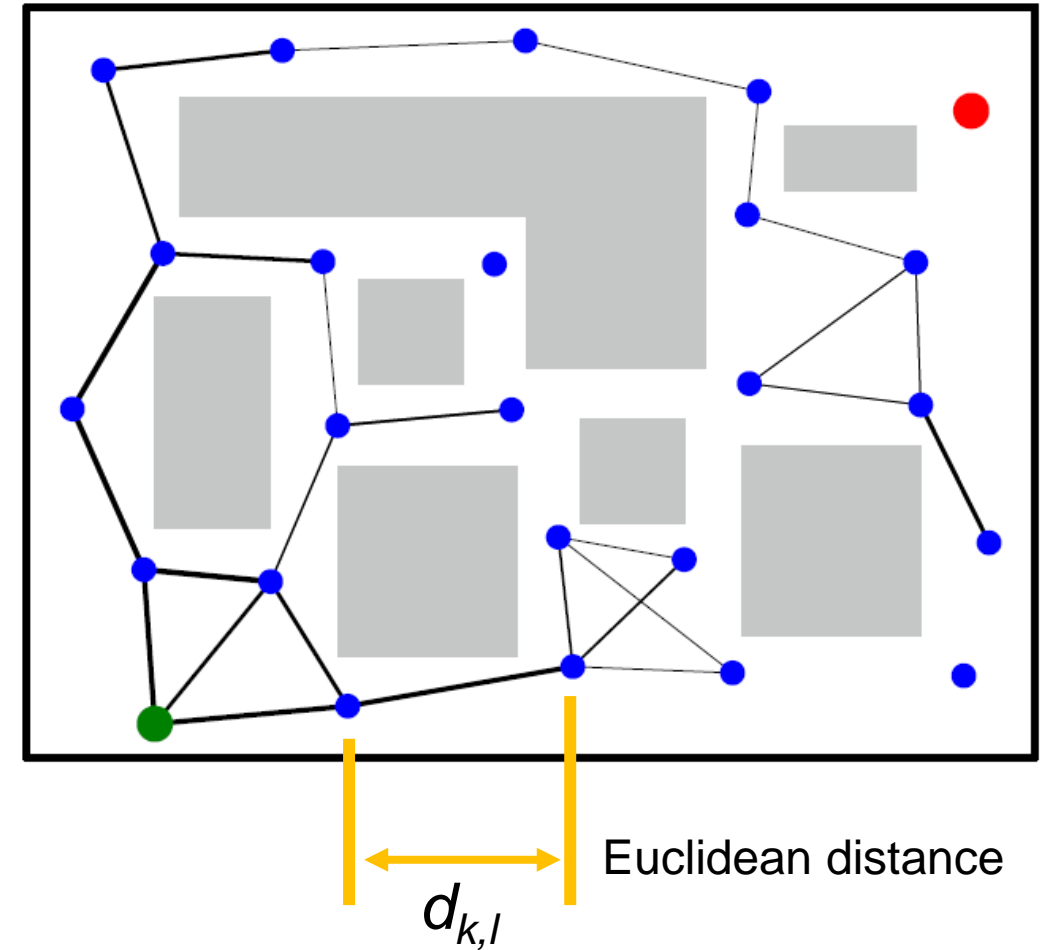
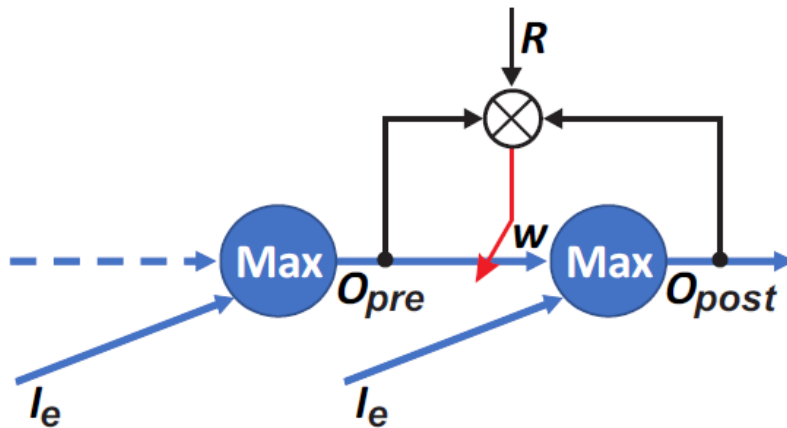
# Navigation learning task

- **Task definition:**
  - Explore environment and find shortest paths



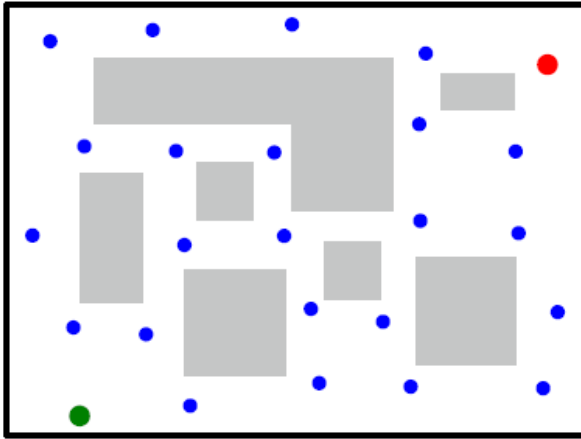
# Navigation learning task

- **Task definition:**
  - Explore environment and find shortest paths
- **Learning rule:**
  - Weight change:  $\Delta w = \alpha O_{pre} O_{post} R,$
  - Reward:  $R = 1 - \beta d_{k,l}$

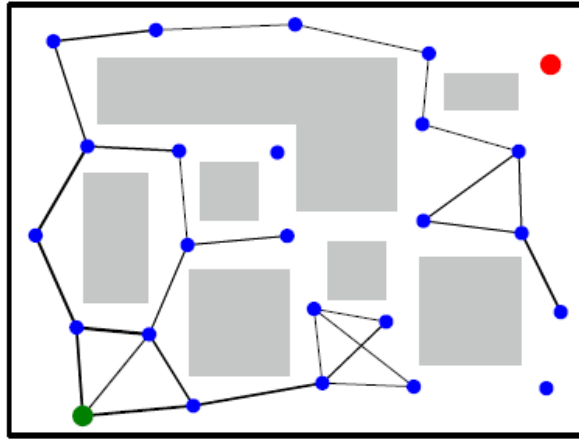


# Navigation learning: Case 1

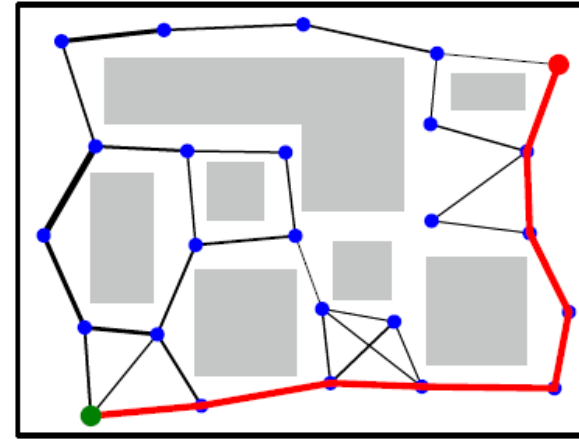
Iteration 0



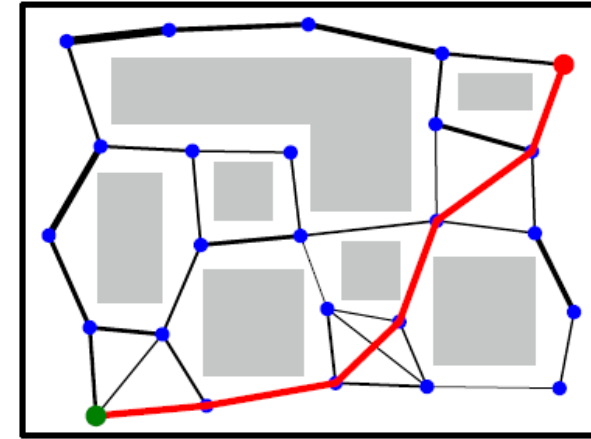
Iteration 100



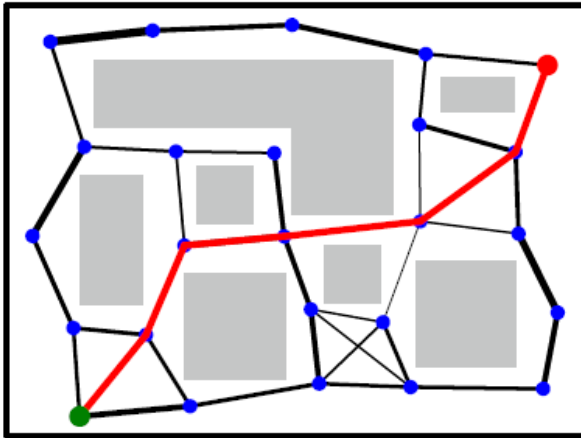
Iteration 200



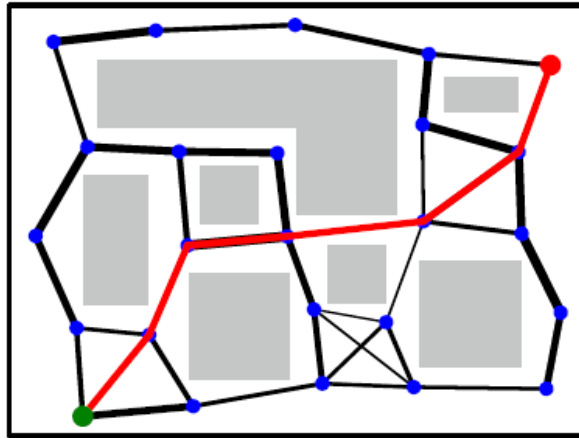
Iteration 300



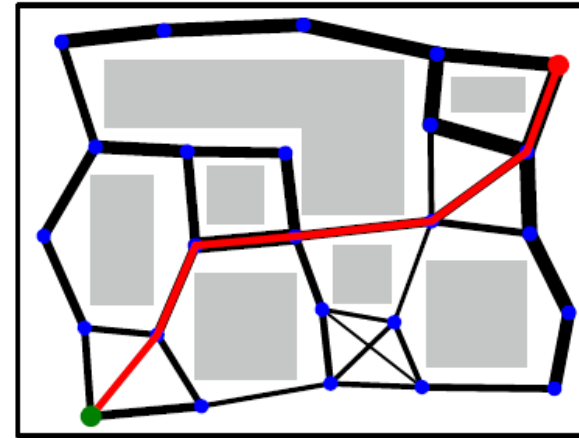
Iteration 400



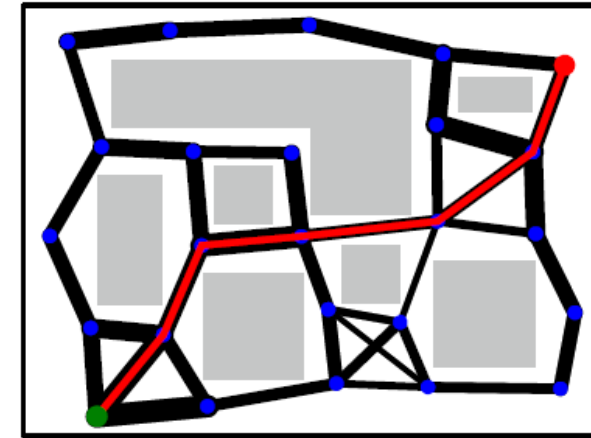
Iteration 600



Iteration 1000



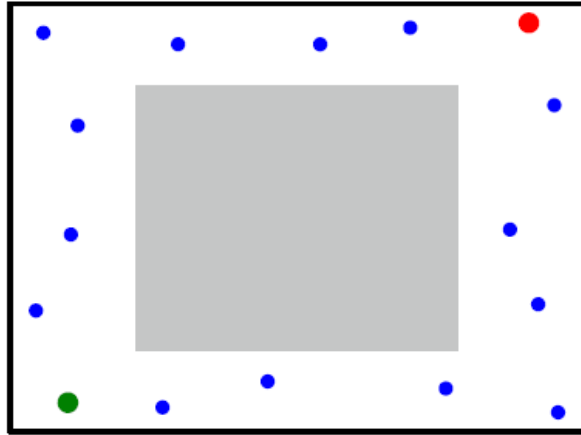
Iteration 1500



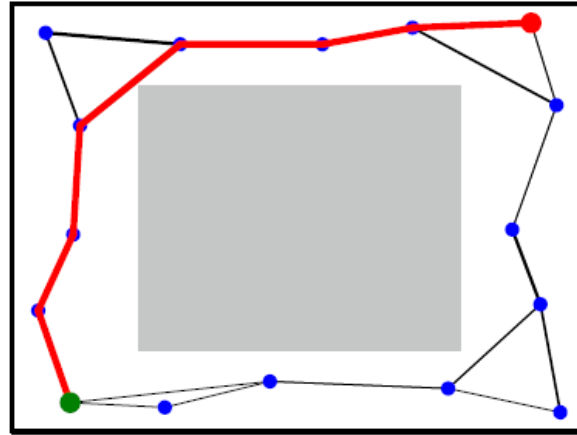


# Navigation learning: Case 2

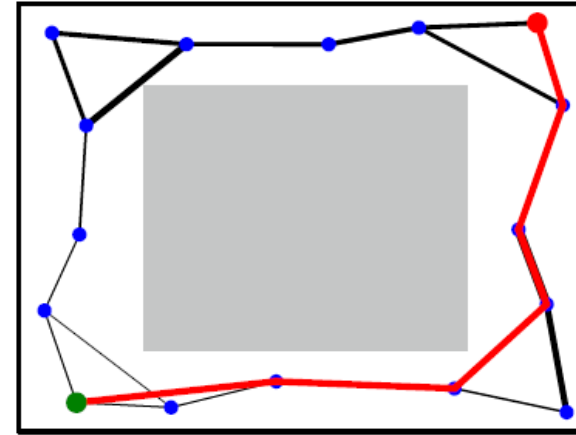
Iteration 0



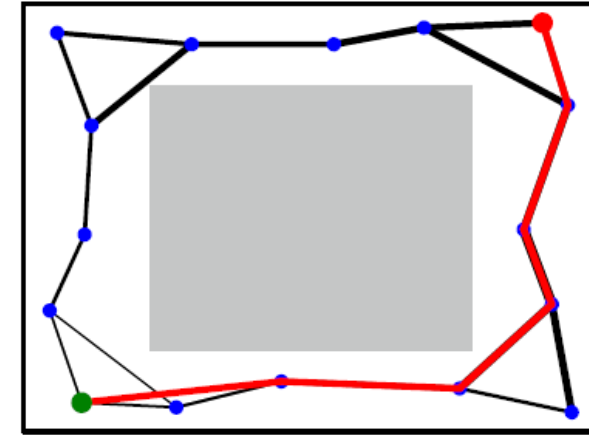
Iteration 100



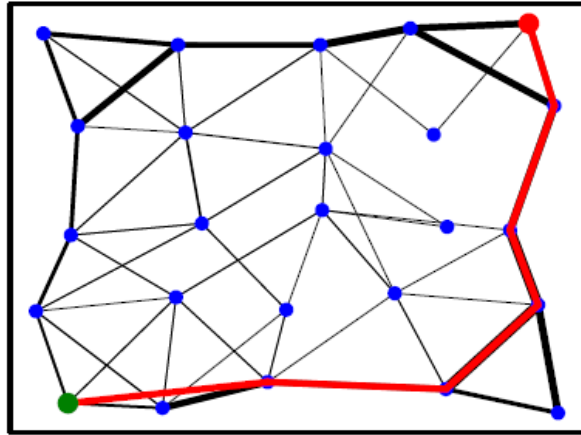
Iteration 200



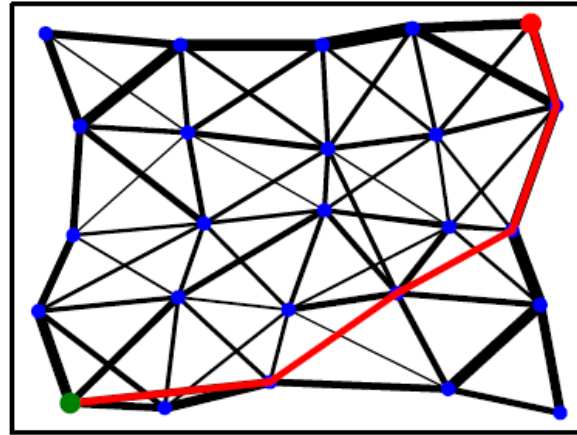
Iteration 300



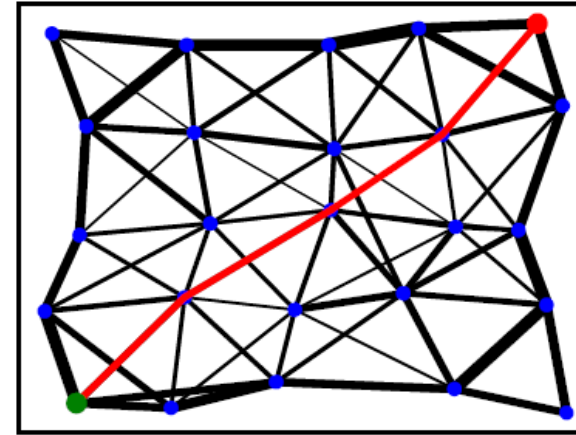
Iteration 400



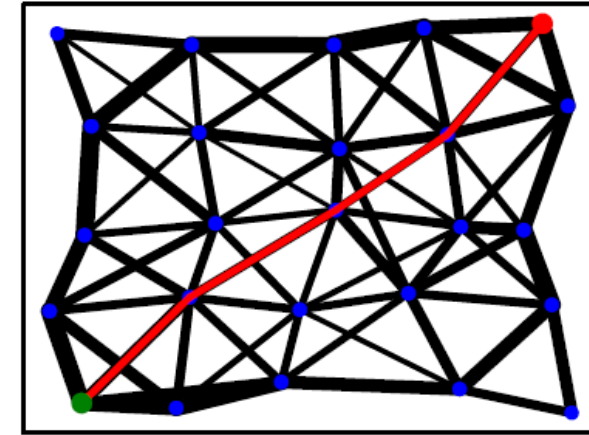
Iteration 1100



Iteration 1200

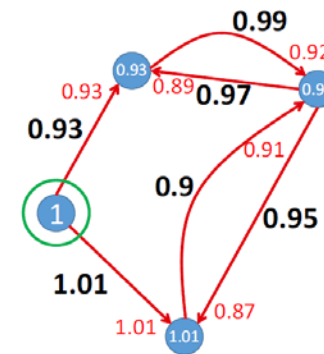
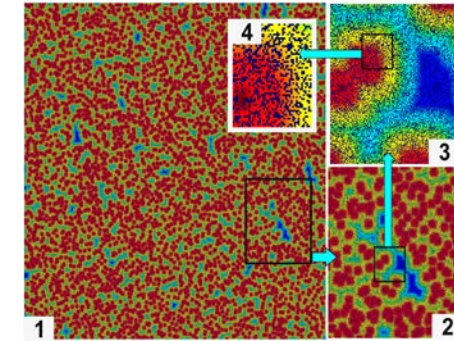
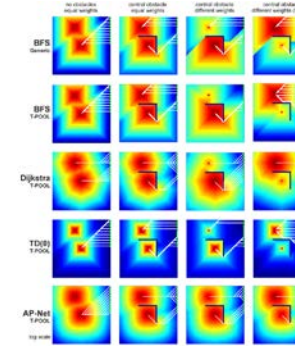


Iteration 2000

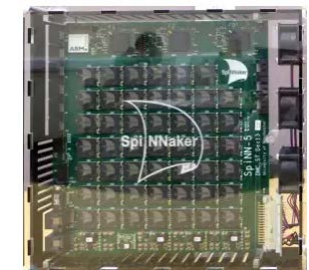
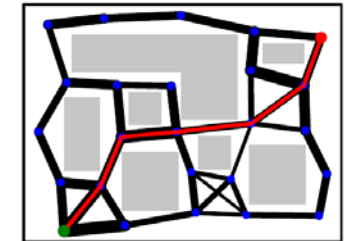


# Summary

- **Deep network without learning**
  - Optimal solutions
  - Only positive costs
- **Neural implementation of BF algorithm**
  - Optimal solutions
  - Positive and negative costs
- **Combination of optimal planning and learning**
- **Potential to apply on neuromorphic hardware**



Hebbian learning



# Thank you!

Florentin Worgötter  
Minija Tamosiunaite  
Sebastian Herzog

- Kulvicius, T., Tamosiunaite, M., & Wörgötter, F. (2023). Combining optimal path search with task-dependent learning in a neural network. *IEEE Transactions on Neural Networks and Learning Systems*. DOI:10.1109/TNNLS.2023.3327103
- Kulvicius, T., Herzog, S., Tamosiunaite, M., & Wörgötter, F. (2021). Finding Optimal Paths Using Networks Without Learning--Unifying Classical Approaches. *IEEE Transactions on Neural Networks and Learning Systems*. DOI:10.1109/TNNLS.2021.3089023

